MuL App Engine Recipes Documentation

Release 0.4.0

Michael Lenaghan

June 01, 2015

Contents

Contents:

CHAPTER 1

MuL App Engine Recipes

Buildout recipes for App Engine.

CHAPTER 2

App Engine SDK Recipe

Given a URL pointing to a Google App Engine Python SDK zip file, this recipe a) downloads the file and b) unzips the file and c) creates top-level Python scripts that invoke corresponding top-level SDK scripts.

Downloading uses and respects the options in the `buildout.cfg`'s `buildout` section. Specifically, downloaded files are cached in the `download-cache` directory and files are never downloaded in `offline` mode.

The downloaded file (cached or not) is unzipped into the recipe's part directory. The top level of the SDK directory is then scanned for Python scripts. For each SDK script a corresponding script is generated and placed in buildout's `bin-directory`. The generated script simply invokes the SDK script, passing along arguments. Buildout's `relative-paths` option is respected. Or, at least, it should be.

The list of generated scripts can be filtered using the `scripts` option. By default its value is '*'. If a '*' appears anywhere in the list all scripts will be generated. Otherwise, only those scripts mentioned in the option (with or without a '.py' extension) will be generated.

2.1 Options

scripts A space-delimited list of Python script names or '*'. The default is '*'.

url The url to the Google App Engine Python SDK zip file. Required.

2.2 Example

```
[sdk]
recipe = mul.recipe.appengine:sdk

url = \
    https://storage.googleapis.com/appengine-sdks/featured/google_appengine_1.9.21.zip
```

CHAPTER 3

App Engine Lib Recipe

Given a list of eggs, this recipe a) computes their working set and then b) copies that working set into the specified `lib-directory`. (This recipe only copies eggs; it doesn't download or install them.) The `lib-directory` is created each time the part is installed and deleted each time the part is uninstalled.

Eggs are copied unless they're in the `ignore-eggs` list. The top-level packages of each egg are copied unless they're in the `ignore-packages` list. The files and directories of each package are copied unless they're in the `ignore-files` list. The `ignore-files` list supports globs. You must, of course, filter packages and files with care; eggs aren't written to expect that kind of install-time surgery.

When an egg is copied its egg-info is also copied. The egg-info can be used for example by pkg_resources to locate package resources at runtime. The egg-info is copied in a manor similar to `setup.py`'s `--single-version-externally-managed` install option; the egg-info directories are siblings of the package directories.

The eggs that are copied can be either zipped or unzipped.

3.1 Options

eggs A newline-delimited list of eggs to copy. The default is an empty list.

ignore-eggs A newline-delimited list of eggs to ignore when copying. The default is an empty list.

ignore-packages A newline-delimited list of packages to ignore when copying. The default is an empty list.

ignore-files A newline-delimited list of file globs to ignore when copying. The default is an empty list.

lib-directory The directory to copy the egg-info and packages to. Required.

3.2 Example

```
[lib]
recipe = mul.recipe.appengine:lib

eggs =
    pyramid
    pyramid_debugtoolbar
ignore-eggs =
    MyEgg
ignore-packages =
```

```
    easy_install
    setuptools
    site
ignore-files =
    *.c
    *.h
    *.pyc
    *.pyo
    *.so
    test
    tests
    testsuite
lib-directory = develop/MyEgg/lib
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/michaellenaghan/mul.recipe.appengine/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

MuL App Engine Recipes could always use more documentation, whether as part of the official MuL App Engine Recipes docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/michaellenaghan/mul.recipe.appengine/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *mul.recipe.appengine* for local development.

1. Fork the *mul.recipe.appengine* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/mul.recipe.appengine.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv mul.recipe.appengine
$ cd mul.recipe.appengine/
$ ./bin/python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 mul.recipe.appengine tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for all supported Python versions. Check https://travis-ci.org/michaellenaghan/mul.recipe.appengine/pull_requests and make sure that the tests pass.

CHAPTER 5

Credits

5.1 Developer

Michael Lenaghan <metamul -@- gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.4.0 (2015-06-01)

- Fixes & Simplifications.

6.2 0.3.0 (2015-05-31)

- Configuration tweaks.

6.3 0.2.0 (2015-05-31)

- Documentation tweaks.

6.4 0.1.0 (2015-05-31)

- First release on PyPI.

CHAPTER 7

License